XEP-0357: Push Notifications

Kevin Smith
mailto:kevin@kismith.co.uk
xmpp:kevin@doomsong.co.uk

Lance Stout
mailto:lance@andyet.com
xmpp:lance@lance.im

2018-10-01
Version 0.4.0

Status	Type	Short Name
Experimental	Standards Track	push

This specification defines a way for an XMPP servers to deliver information for use in push notifications to mobile and other devices.

Legal

Copyright

Permissions

Warranty

Liability

Conformance

Contents

1 Introduction

The purpose of push notifications is to inform users of new messages or other pertinent information even when they have no XMPP clients online.
Typically, these notifications are delivered to a user's mobile device, displaying a notice that can trigger opening an XMPP client to continue a conversation or answer a Jingle session request.
There have been several push notification implementations by mobile XMPP client vendors. However, experience has shown that these implementations carried several drawbacks:

- Treated the XMPP client and XMPP server as one unified service, such that push notifications only worked using the "official" client.
- Proxied a user's session through the client provider's backend services in order to monitor for and trigger push notifications.

The goal for this document is to make the generalized case possible, whereby a user may use their XMPP client of choice with their own server of choice. The requirements are thus:

- Allow XMPP servers to support push notifications to multiple client implementations, via multiple external or proprietary push services.
- Allow clients to receive push notifications from multiple third-party XMPP servers.
- Eliminate the need for clients to proxy a user's XMPP session in order to enable push notifications.

Note: Any publish-subscribe use cases not described herein are described in Publish-Subscribe (XEP-0060) [1]. Also, this document does not show error flows related to the generic publish-subscribe use cases referenced herein, since they are exhaustively defined in XEP-0060. The reader is referred to XEP-0060 for all relevant protocol details related to the XMPP publish-subscribe extension. This document merely defines a "subset" or "profile" of XMPP publish-subscribe.

2 Concepts and Approach

XMPP Push works between the user's XMPP server and two push notification services in tandem:

1. The user's XMPP server publishes notifications to the XMPP Push Service of each of the user's client applications.

[1] XEP-0060: Publish-Subscribe <https://xmpp.org/extensions/xep-0060.html>.

2. The XMPP Push Service (as defined here) for a client application then delivers the notification to a third-party notification delivery service.
3. The third-party (and potentially proprietary or platform-dependent) push service delivers the notification from the client application's backend service to the user's device.

This two-tiered push architecture allows the user's XMPP server to deliver notifications to arbitrary third-party clients, and in turn allows those clients to use the appropriate delivery mechanism for their platforms without having to share any private keys or other credentials with the XMPP server.

2.1 General Architecture of a Push Notification Service

The current state-of-the-art for a generic push notification service requires four actors:

App Client The app client is the software installed and ran by the user, and is the final receiver of a push notification.

App Server The app server is a backend service for the app client. At minimum, the app server exists to trigger push notifications, but it often also performs business logic for the app.

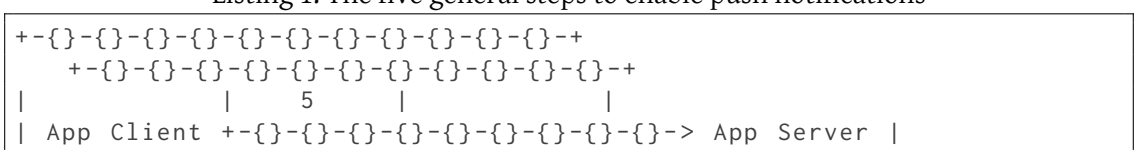
User Agent The user agent is a service running locally on the user's device which receives push notifications and delivers them to the appropriate application.

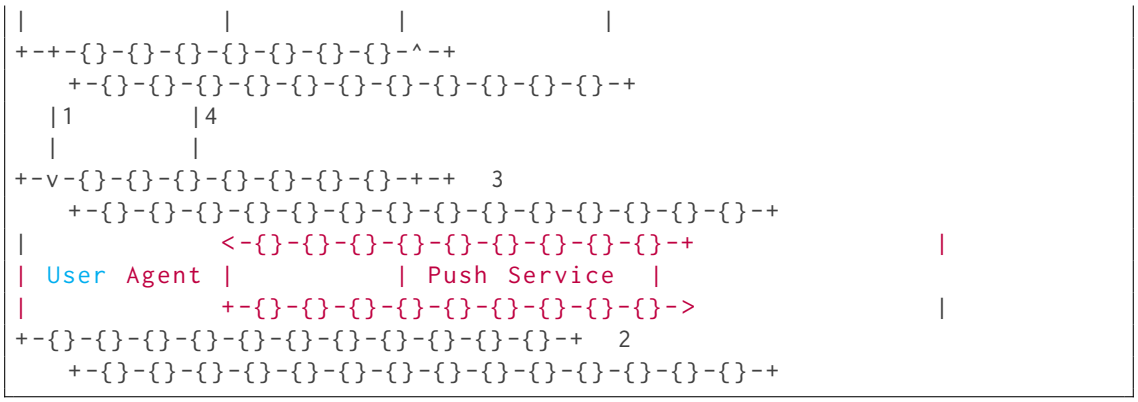
Push Service The push service ferries notifications from the App Server to the User Agent. How it does so is often proprietary and vendor/platform dependent.

Enabling notifications is a five step process:

1. The App Client asks the User Agent to authorize the delivery of notifications.
2. The User Agent then requests a token from the Push Service which authorizes delivery of notifications to that User Agent and App Client.
3. The Push Service issues the token to the User Agent.
4. The User Agent gives the token to the App Client.
5. The App Client sends the token to the App Server for later use.

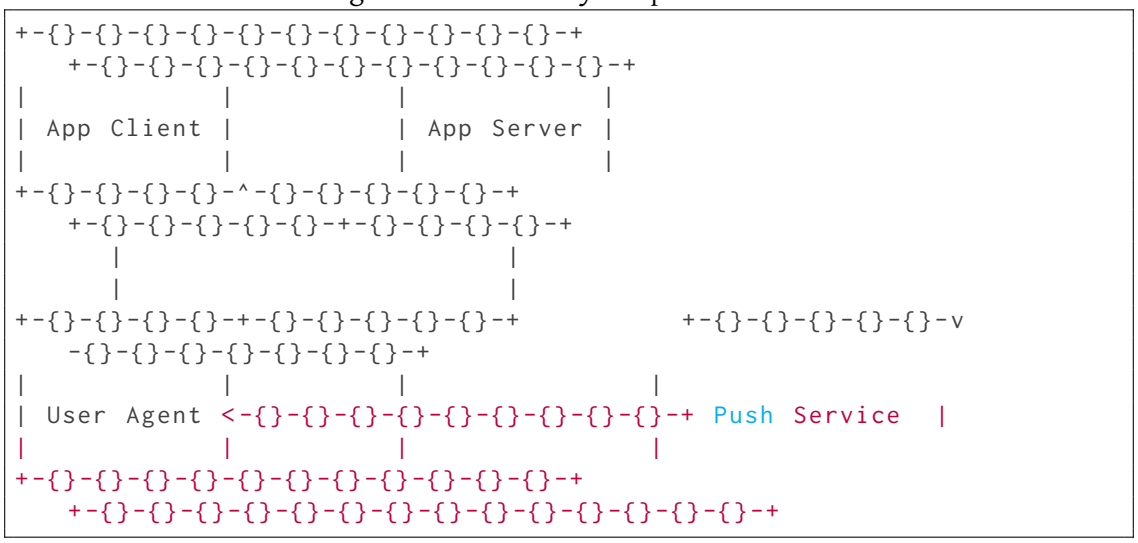
Listing 1: The five general steps to enable push notifications





To send a push notification, the App Server sends the notification data to the Push Service along with the saved token.

Listing 2: General delivery of a push notification

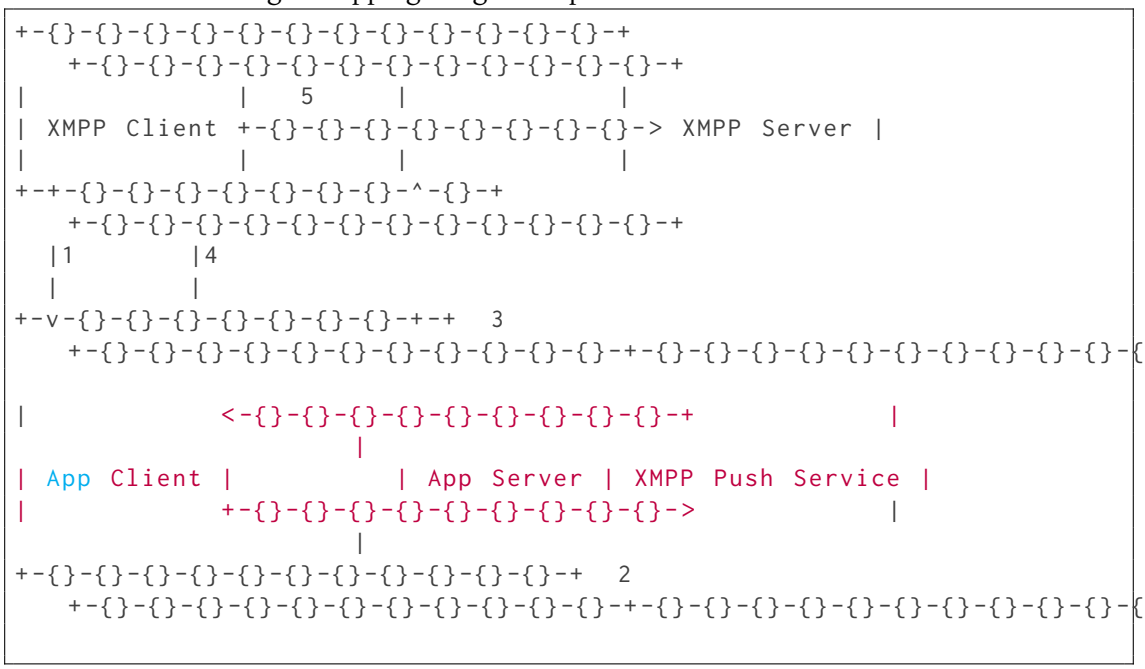


2.2 Mapping the General Architecture to XMPP

To build an XMPP Push service on top of a general push service, we perform the following mapping:

- The general App Client becomes the XMPP User Agent
- The general App Server becomes the XMPP Push Service
- The XMPP server is now the new logical ”App Server”
- The XMPP client portion of the application is the new logical ”App Client”

Listing 3: Mapping the generic push architecture to use XMPP



3 XMPP Push Service

An XMPP Push Service is a PubSub service as defined by the XMPP XEP-0060 extension. The functional difference between a Push Service and a generic pubsub service is that a Push Service will generally summarize and forward published content via non-XMPP mechanisms. Note: a Push Service is provided by a specific client application as part of the App Server. A user's XMPP server will typically *not* act as a Push Service itself, but will instead publish to the Push Services for the user's client applications.

3.1 Recommended Defaults

A Push Service MUST:

- Support the 'whitelist' access model and set it to the default.
- Support the 'publish-only' affiliation.

3.2 Business Rules

Each PubSub node is a delivery target for the Push Service, which could represent multiple devices for a single user.
In order to prevent information leaks, each node SHOULD be configured with a 'whitelist'

access model so that only trusted entities are able to view or subscribe to published notifications. Furthermore, the 'publish-only' affiliation SHOULD be used to allow acceptable entities (such as the server JID and the user's bare JID) to publish to the node to trigger notifications. Care SHOULD be taken to ensure that publish requests are coming from the user's server and not from other third-party client applications using the full JID of a user. A Push Service MAY opt to only accept or further process publish requests from server JIDs and bare user JIDs to ensure that only a user's server is able to publish, but it SHOULD instead use publish options with credentials shared only with the user's server (see Enabling Notifications).

4 Discovering Support

4.1 Account Owner Service Discovery

Before enabling or disabling push services, a client SHOULD determine whether the user's server supports publishing push notifications; to do so, it MUST send a Service Discovery (XEP-0030) [2] information quest to the user's bare JID:

Listing 4: Client queries server regarding protocol support

```
<iq from='user@example.com/mobile'
    to='user@example.com'
    id='x13'
    type='get'>
  <query xmlns='http://jabber.org/protocol/disco#info'/>
</iq>
```

If the user's server supports publishing push notifications and the account is provisioned to allow them, the server MUST include the feature 'urn:xmpp:push:0' in its list of supported features.

Listing 5: Server communicates protocol support

```
<iq from='juliet@capulet.lit'
    to='juliet@capulet.lit/balcony'
    id='disco1'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category='account' type='registered'/>
    <feature var='urn:xmpp:push:0'/>
    ...
  </query>
</iq>
```

[2] XEP-0030: Service Discovery <https://xmpp.org/extensions/xep-0030.html>.

4.2 Push Service Discovery

If a service supports the XMPP Push Service publish-subscribe profile described herein, it MUST include an identity of ”pubsub/push” in ”disco#info” results.

Listing 6: Service identifies as a Push Services

```
<iq from='push-5.client.example'
    to='user@example.com/mobile'
    id='x23'
    type='result'>
  <query xmlns='http://jabber.org/protocol/disco#info'>
    <identity category='pubsub' type='push' />
    <feature var='urn:xmpp:push:0'/>
    ...
  </query>
</iq>
```

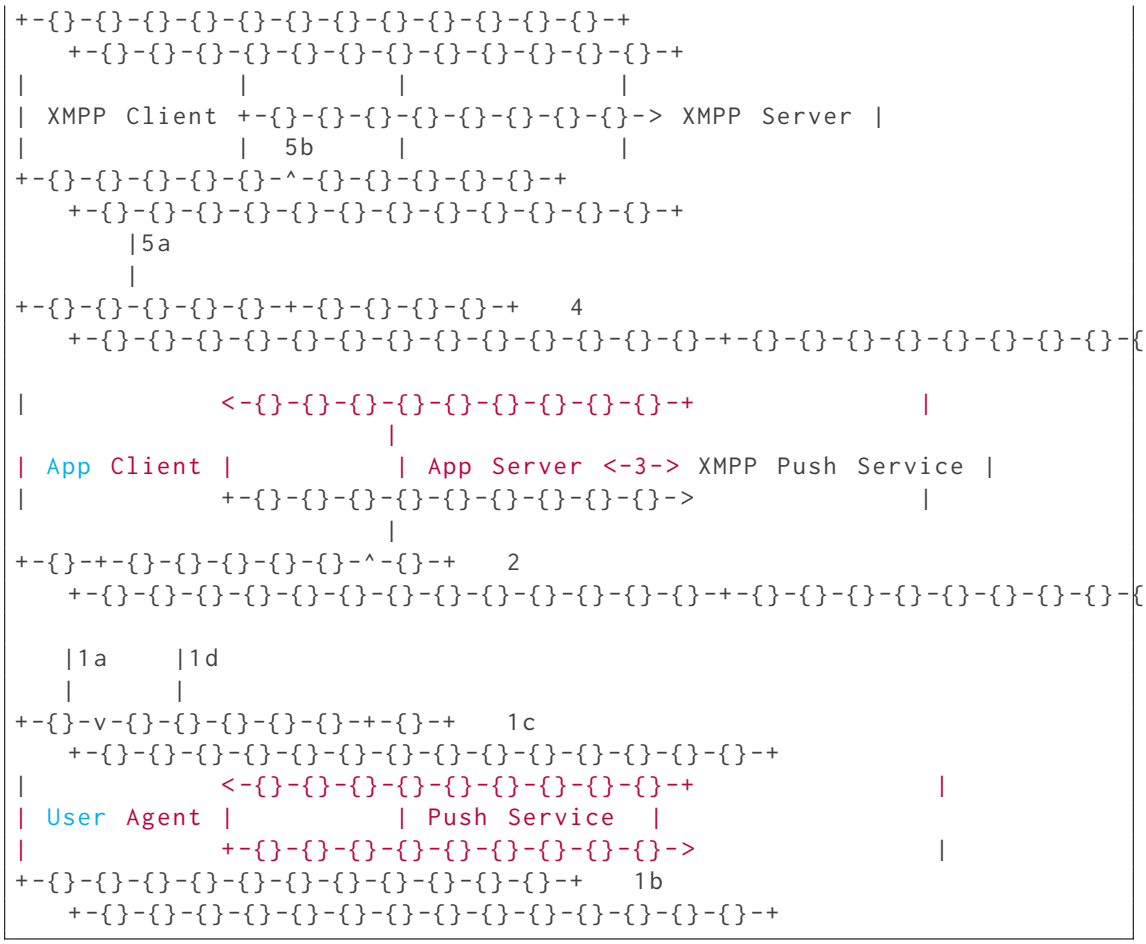
5 Enabling Notifications

The full process for enabling notifications requires initializing two separate push services: between the App Client and App Server, and between the App Server and the user's XMPP server.
Note: It is assumed that an App Client is able to perform any registration procedures it requires to bootstrap its own preferred push notification system. Furthermore, it is assumed that the App Client or App Server is able to provision a node on its own XMPP Push Service. It is possible, but not required, to perform these actions over XMPP using In-Band Registration (XEP-0077) [3].

1. The App Client performs any necessary bootstrapping and registration for its preferred push service.
2. The App Client registers itself with the App Server.
3. The App Server allocates or reuses a node on the App Server's XMPP Push Service.
4. The App Server informs the App Client of the provisioned node, along with any additional parameters required for publishing to that node.
5. The App Client requests the XMPP server to publish notifications to the given node.

Listing 7: The full flow of enabling push notifications for an application

[3] XEP-0077: In-Band Registration <https://xmpp.org/extensions/xep-0077.html>.



For the last step, the App Client sends an IQ-set to the user's bare JID with an <enable /> element qualified by the 'urn:xmpp:push:0' namespace, which MUST contain a 'jid' attribute of the XMPP Push Service being enabled. It SHOULD contain a 'node' attribute which is set to the provisioned node specified by the App Server.

Listing 8: Enabling Notifications

```
<iq type='set' id='x42'>
  <enable xmlns='urn:xmpp:push:0' jid='push-5.client.example' node='
     yxs32uqsflafdk3iuqo' />
</iq>
```

An App Server MAY require additional information to be provided with each published notification, such as authentication credentials. These parameters are included in the enable request by adding a Data Forms (XEP-0004) [4] data form with a FORM_TYPE of 'http://jabber.org/protocol/pubsub#publish-options'.

[4] XEP-0004: Data Forms <https://xmpp.org/extensions/xep-0004.html>.

Listing 9: Enabling Notifications, with provided publish options

```
<iq type='set' id='x43'>
  <enable xmlns='urn:xmpp:push:0' jid='push-5.client.example' node='
      yxs32uqsflafdk3iuqo'>
    <x xmlns='jabber:x:data' type='submit'>
      <field var='FORM_TYPE'><value>http://jabber.org/protocol/pubsub#
          publish-options</value></field>
      <field var='secret'><value>eruio234vzxc2kla-91</value></field>
    </x>
  </enable>
</iq>
```

The JID for a Push Service MAY be enabled multiple times for a user only if different node values are provided. If the combination of JID and node has already been enabled, then the server SHOULD use the last received request for any publish options.

6 Disabling Notifications

If the user decides to stop push notifications for a particular client application, the App Client SHOULD send an IQ-set to the user's bare JID with a <disable /> element qualified by the 'urn:xmpp:push:0' namespace, which MUST include a 'jid' attribute of the service to be removed.

Listing 10: Disabling all notifications to a given service

```
<iq type='set' id='x97'>
  <disable xmlns='urn:xmpp:push:0' jid='push-5.client.example' />
</iq>
```

A 'node' attribute MAY be included to remove a particular JID and node combination if multiple nodes have been enabled for a single service JID.

Listing 11: Disabling notifications

```
<iq type='set' id='x97'>
  <disable xmlns='urn:xmpp:push:0' jid='push-5.client.example' node='
      yxs32uqsflafdk3iuqo' />
</iq>
```

If a 'node' attribute is provided, then only that combination of JID and node SHOULD be removed from the set of enabled services. Otherwise, the server SHOULD disable all enabled entries for the specified service for the user.
When a service is not enabled, the server MUST NOT attempt publishing notifications to the service.

7 Publishing Notifications

When the user's server detects an event warranting a push notification, it performs a PubSub publish to all XMPP Push Services registered for the user, where the item payload is a <notification /> element in the 'urn:xmpp:push:0' namespace.
A Data Forms (XEP-0004) [5] data form whose FORM_TYPE is 'urn:xmpp:push:summary' MAY be included to provide summarized information such as the number of unread messages or number of pending subscription requests.
Other elements MAY be included if relevant for the notification.

Listing 12: Server publishes a push notification

```
<iq type='set'
    from='example.com'
    to='push-5.client.example'
    id='n12'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='yxs32uqsflafdk3iuqo'>
      <item>
        <notification xmlns='urn:xmpp:push:0'>
          <x xmlns='jabber:x:data'>
            <field var='FORM_TYPE'><value>urn:xmpp:push:summary</value
                ></field>
            <field var='message-count'><value>1</value></field>
            <field var='last-message-sender'><value>juliet@capulet.
                example/balcony</value></field>
            <field var='last-message-body'><value>Wherefore art thou,
                Romeo?</value></field>
          </x>
          <additional xmlns='http://example.com/custom'>Additional
              custom elements</additional>
        </notification>
      </item>
    </publish>
  </pubsub>
</iq>
```

If additional data was provided when enabling the service, the publish request SHOULD include the data as publish options.

Listing 13: Server publishes a push notification with provided publish options

```
<iq type='set'
    from='example.com'
    to='push-5.client.example'
    id='n12'>
```

[5] XEP-0004: Data Forms <https://xmpp.org/extensions/xep-0004.html>.

```
  <pubsub xmlns='http://jabber.org/protocol/pubsub'>
    <publish node='yxs32uqsflafdk3iuqo'>
      <item>
        <notification xmlns='urn:xmpp:push:0'>
          <x xmlns='jabber:x:data'>
            <field var='FORM_TYPE'><value>urn:xmpp:push:summary</value
                ></field>
            <field var='message-count'><value>1</value></field>
            <field var='last-message-sender'><value>juliet@capulet.
                example/balcony</value></field>
            <field var='last-message-body'><value>Wherefore art thou,
                Romeo?</value></field>
          </x>
          <additional xmlns='http://example.com/custom'>Additional
              custom elements</additional>
        </notification>
      </item>
    </publish>
    <publish-options>
      <x xmlns='jabber:x:data'>
        <field var='FORM_TYPE'><value>http://jabber.org/protocol/
            pubsub#publish-options</value></field>
        <field var='secret'><value>eruio234vzxc2kla-91<value></field>
      </x>
    </publish-options>
  </pubsub>
</iq>
```

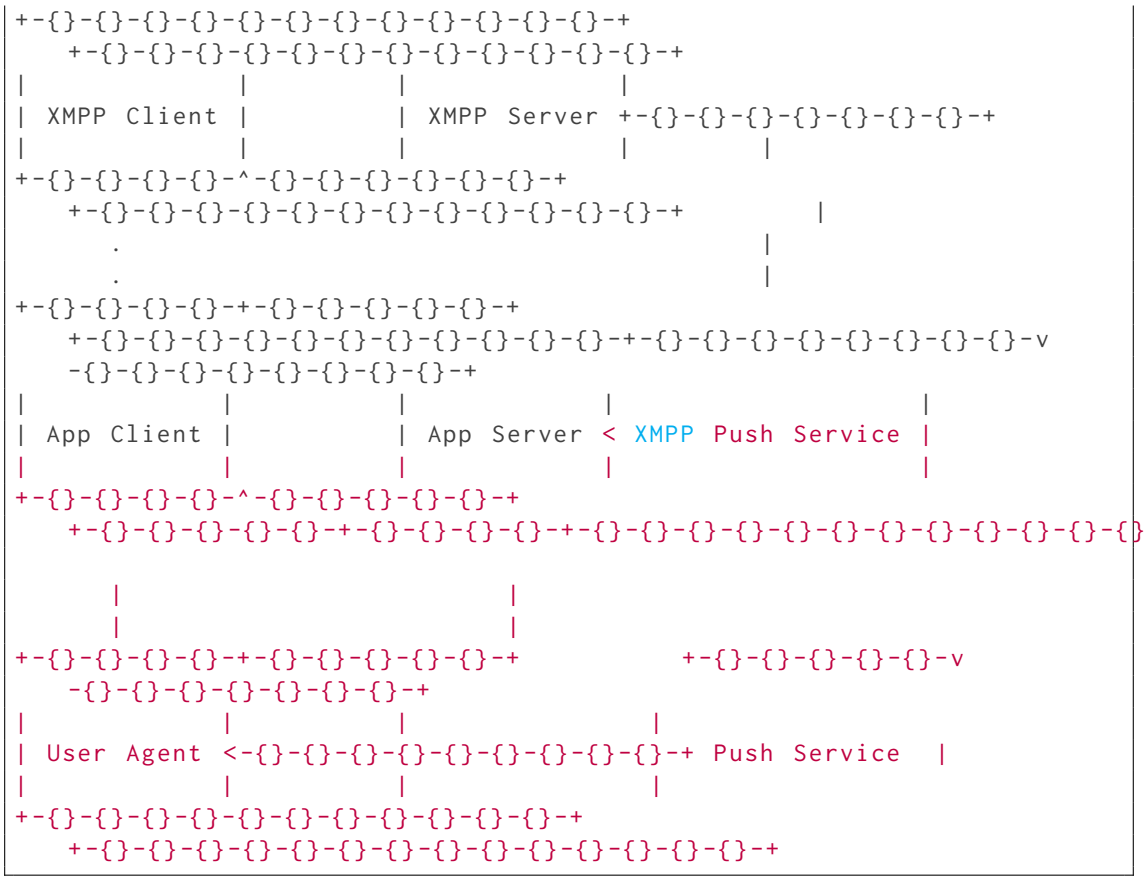
7.1 Publish Errors

If a publish request is returned with an IQ-error, then the server SHOULD consider the particular JID and node combination to be disabled.
However, a server MAY choose to keep a service enabled if the error is deemed recoverable or transient, until a sufficient number of errors have been received in a row.
A server MAY retry an automatically disabled JID and node combination after a period of time (e.g. 1 day).

7.2 Notification Delivery

Once the notification has been published to the XMPP Push Service, it is left to the implementation how to deliver the notification to the user's device. However, the general flow for the process looks like so:

Listing 14: The full path of a push notification, from XMPP server to user client



8 Remote Disabling of Notifications

It can be desirable for an XMPP Push Service to stop accepting notifications from the user's XMPP server. To do so, the XMPP Push Service removes the 'publish-only' (or other publish-enabling affiliation) from the user's JID, and MAY send an affiliation change notice to the user's bare JID:

Listing 15: Push Service announces stop of push support

```
<message from='push-5.client.example' to='user@example.com'>
  <pubsub xmlns='http://jabber.org/protocol/pubsub' node='
      yxs32uqsflafdk3iuqo'>
    <affiliation jid='user@example.com' affiliation='none' />
  </pubsub>
</message>
```

Upon receiving an affiliation change event, the server MAY remove the received JID and node combination from the set of enabled services. If a server does not do so, then the service will

be removed from the enabled set through the error handling process.

9 Security Considerations

Push notifications require routing private information, such as message bodies, through third parties. As such, servers SHOULD allow users to limit the information sent via push notifications.
It is NOT RECOMMENDED to allow in-band modification of push notification content settings. Such operations SHOULD be done out-of-band to prevent privilege escalation.

10 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) [6].

11 XMPP Registrar Considerations

11.1 Protocol Namespaces

The XMPP Registrar [7] includes 'urn:xmpp:push:0' in its registry of protocol namespaces (see <https://xmpp.org/registrar/namespaces.html>).

- urn:xmpp:push:0

11.2 Protocol Versioning

If the protocol defined in this specification undergoes a revision that is not fully backwards-compatible with an older version, the XMPP Registrar shall increment the protocol version number found at the end of the XML namespaces defined herein, as described in Section 4 of XEP-0053.

[6] The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.

[7] The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <https://xmpp.org/registrar/>.

11.3 Field Standardization

Field Standardization for Data Forms (XEP-0068) [8] defines a process for standardizing the fields used within Data Forms scoped by a particular namespace, and the XMPP Registrar maintains a registry of such FORM_TYPES (see <https://xmpp.org/registrar/formtypes.html>).

11.3.1 urn:xmpp:push:summary FORM_TYPE

```
<form_type>
  <name>urn:xmpp:push:summary</name>
  <doc>XEP-XXXX</doc>
  <desc>Provides summarizing information about a user for use in push
     notifications.</desc>
  <field
      var='message-count'
      type='text-single'
      label='The number of unread or undelivered messages'/>
  <field
      var='pending-subscription-count'
      type='text-single'
      label='The number of pending incoming presence subscription
         requests'/>
  <field
      var='last-message-sender'
      type='jid-single'
      label='The sender of the last received message'/>
  <field
      var='last-message-body'
      type='text-single'
      label='The body text of the last received message'/>
</form_type>
```

11.4 Service Discovery Category/Type

The XMPP Registrar includes a category of "component" in its registry of Service Discovery identities (see <https://xmpp.org/registrar/disco-categories.html>); as a result of this document, the Registrar includes a type of "jidprep" to that category.
The registry submission is as follows:

```
<category>
  <name>pubsub</name>
  <type>
    <name>push</name>
    <desc>
```

[8] XEP-0068: Field Data Standardization for Data Forms <https://xmpp.org/extensions/xep-0068.html>.

```
      A push notification service that supports the publish-subscribe
          profile defined in XEP-XXXX.
    </desc>
    <doc>XEP-XXXX</doc>
  </type>
</category>
```

12 XML Schema

```
<?xml version='1.0' encoding='UTF-8'?>

<xs:schema
    xmlns:xs='http://www.w3.org/2001/XMLSchema'
    targetNamespace='urn:xmpp:push:0'
    xmlns='urn:xmpp:push:0'
    elementFormDefault='qualified'>

  <xs:annotation>
    <xs:documentation>
      The protocol documented by this schema is defined in
      XEP-xxxx: http://www.xmpp.org/extensions/xep-xxxx.html
    </xs:documentation>
  </xs:annotation>

  <xs:import
      namespace='jabber:x:data'
      schemaLocation='http://xmpp.org/schemas/x-data.xsd' />

  <xs:element name='enable'>
    <xs:complexType>
      <xs:sequence minOccurs='0' maxOccurs='unbounded' xmlns:xdata='
          jabber:x:data'>
        <xs:element ref='xdata:x' />
      </xs:sequence>
      <xs:attribute name='jid' type='xs:string' use='required' />
      <xs:attribute name='node' type='xs:string' use='required' />
    </xs:complexType>
  </xs:element>

  <xs:element name='disable'>
    <xs:complexType>
      <xs:attribute name='jid' type='xs:string' use='required' />
      <xs:attribute name='node' type='xs:string' use='optional' />
    </xs:complexType>
  </xs:element>

  <xs:element name='notification'>
```

```
    <xs:complexType>
       <xs:sequence minOccurs='0' maxOccurs='unbounded' xmlns:xdata='
           jabber:x:data'>
         <xs:element ref='xdata:x' />
         <xs:any />
       </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
```